IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

U.S. PATENT APPLICATION

FOR:

MOBILE WEB SERVICES

INVENTOR:

PETRI NYKÄNEN

Morgan & Finnegan L.L.P.
345 Park Avenue, 22nd Floor
New York, NY 10154-0053
(212) 758-4800 (Telephone)
(212) 751-6849 (Facsimile)

1775 Eye Street, N.W., Suite 400
Washington D.C., 20006
(202) 857-7887 (Telephone)
(202) 857-7929 (Facsimile)

Attorneys For Applicant

MOBILE WEB SERVICES

BACKGROUND OF THE INVENTION

Field of the Invention:

The invention disclosed broadly relates to methods for providing Internet services and more particularly relates to improvements in mobile device accessing of Internet services.

Background Art:

Universal Description, Discovery and Integration (UDDI) is a defacto standard for an Internet-based registry. The UDDI registry enables users to discover services and businesses on the Internet. The UDDI standard takes advantage of WorldWide Web Consortium (W3C) standards and Internet Engineering Task Force (IETF) standards such as Extensible Markup Language (XML), Hypertext Transfer Protocol (HTTP), Domain Name System (DNS) protocol, and Simple Object Access Protocol (SOAP) messaging protocol. The UDDI registry enables users to quickly, easily and dynamically find businesses and services on the Internet. The UDDI registry enables businesses to reach their customers and partners with information about their products and Web services. The UDDI registry also enables businesses to integrate into each other's systems and processes. Registering enables a business to publicly list basic information about its company and offerings. There is also the option to list a catalog of products, services and guidelines for engagement. Registered businesses and their catalogs of services and products

are then accessible in searches by potential buyers. Details of the UDDI registry and its searching protocol can be found in the following online papers:

UDDI Executive White Paper, September 6, 2000, http://www.uddi.org/pubs/UDDI_Executive_White_Paper.pdf

UDDI Technical White Paper, September 6, 2000, http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

UDDI Programer's API 1.0, UDDI Open Draft Specification 30 September 2000, by Toufic Boubez, et al., http://www.uddi.org/pubs/ProgrammersAPI-V1-1.pdf

UDDI Data Structure Reference, UDDI Open Draft Specification 30 September 2000, by Toufic Boubez, et al., http://www.uddi.org/pubs/DataStructure-V1.pdf

Mobile phones and wireless personal digital assistants (PDAs) are able to access the Internet using the Wireless Application Protocol (WAP). The Nokia WAP Client Version 2.0 is a software product containing the components necessary to implement a WAP client on a wireless device. These components include a Wireless Markup Language (WML) Browser, WMLScript engine, Push Subsystem, and Wireless Protocol Stack. The Nokia WAP Client is a source-code product that can port and integrate into wireless devices such as mobile phones and wireless PDAs. Application programs stored in the wireless device interact with the WAP Client to implement a variety of communications applications. Details of the Nokia WAP Client

Version 2.0 can be found in the online paper: Nokia WAP Client Version 2.0, Product Overview, Nokia Internet Communications, 2000, www.nokia.com/corporate/wap.

What is needed is the ability of a mobile phone or wireless PDA to discover Internet businesses and services by accessing the UDDI registry. It would be even more useful to facilitate the formation of a query to the UDDI registry for the wireless device user. It would be beneficial to maintain a personal profile of the user's Internet accessing preferences and to use that profile in formulating a detailed UDDI query of the UDDI registry from the user's abbreviated inputs to the wireless device.

SUMMARY OF THE INVENTION:

In accordance with the invention, a method is disclosed to enable a mobile phone or wireless PDA to discover Internet businesses and services by accessing the Universal Description, Discovery and Integration (UDDI) registry. The method facilitates the formation of a query to the UDDI registry for the wireless device user. The method constructs a personal user profile of the user's UDDI searching strategies and Internet accessing preferences. The user profile can be used as a shortcut for online or offline queries to the UDDI registry or for accessing pages from web sites, in response to the user's entry of abbreviated inputs to the wireless device. The method is embodied as programmed instructions which may be executed within the user's wireless device to query the UDDI registry. Alternately, method is embodied as programmed instructions which may be executed within a separate knowledge engine server to query the UDDI registry in response to commands from the user's wireless device. The server

can be used to cache files accessed from web sites, for selective forwarding to the user's wireless device.

In one aspect of the invention, the sequence of operational steps for the wireless device's UDDI registry browsing method begins by entering a search handle that will be associated with the user's search strategy. Then the query terms are entered by the user, which may be a business name, part of a business name, a service description, or other characterization. If the characterization is part of a business name, the wireless device then sends a *find_business* XML inquiry using Simple Object Access Protocol (SOAP) to the UDDI registry. The wireless device then receives back from the UDDI registry, a *businessList* message that contains a list of business names satisfying the *find_business* query. The user can then select an item from the returned *businessList* message and drill-down in the selected business' entity data.

The wireless device then sends a *find_service* XML inquiry using SOAP to the UDDI registry. The wireless device then receives back from the UDDI registry, a *serviceList* message that contains a list of names of services offered by the selected business. The user can then select an item from the returned *serviceList* message and drill-down in the selected service data.

The wireless device then sends a *_get_serviceDetail_* XML inquiry using SOAP to the UDDI registry. The wireless device then receives back from the UDDI registry, a *serviceDetail* message that includes bindingTemplate data that contains the details of the selected service. Included in the *bindingTemplate* data is the *accessPoint URL*, which is the URL of the selected service on the web site of the selected business. The service details may be comprehensive or abbreviated, and may also be prioritized according to the needs of the user (or not prioritized at all).

The wireless device then displays the accessPoint URL to the user. The wireless device also stores the search handle in user profile with the selected accessPoint URL, to enable the user to access web pages from the web site of the selected business. This provides the user with a shortcut for accessing pages from web sites, in response to the user's entry of abbreviated search handle to the wireless device.

The wireless device also stores the search handle in user profile with the UDDI registry search strategy. The stored search strategy includes the business name query , the selected *businessEntity* data, the selected *businessService* data, the selected *bindingTemplate* data, and the selected *accessPoint* URL. This provides the user with a shortcut for online or offline queries to the UDDI registry, in response to the user's entry of abbreviated search handle to the wireless device. The search handle may be associated with an icon on the user's screen. Thus, activation of the icon by the user would initiate the shortcut.

To replay a UDDI registry search strategy, the user merely enters a search handle into the wireless device and selects the replay option. The wireless device then accesses the UDDI registry search strategy from user profile corresponding to the search handle and loads the business name query , the selected *businessEntity* data, the selected *businessService* data, and the selected *bindingTemplate* data as each respective operand that would have been otherwise entered by the user. If the data in the UDDI registry has been updated since the user's last query, the *bindingTemplate* data may contain additional or modified *accessPoint URLs*, of the selected service on the web site of the selected business.

In another aspect of the invention, the sequence of operational steps described above for the wireless device's UDDI registry browsing method can also be carried out in a separate knowledge engine server. The server performs the above described method to query the UDDI

registry in response to commands from the user's wireless device. The knowledge engine in the server begins by receiving user's query or search handle. If a search handle has been received, then the server replays a corresponding search strategy for the UDDI registry accessed from the user's profile and uses the query steps specified in the strategy instead of requesting further inputs from the user.

If, however, the knowledge engine server has received a new user query, the server then accesses the UDDI registry using the method described above to identify web sites. The server returns a list of web sites to the user and stores binding templates in the user's profile. The server then receives the user's selection of web sites to search and the server updates user profile with these preferences.

Whether the server begins by receiving the user's new query or the user's search handle, the server proceeds to search the identified web sites using the URLs contained in the stored binding templates. The server retrieves any documents resulting from the search of the specified web sites. The server applies a filter that is prescribed by the user and stored in the user's profile, to limit the returned documents to only those of particular interest to the user. The server sorts the documents in a list having an order established in accordance with user's profile. The server further stores the filtered documents and the sorted list in a cache for later use. The documents may subsequently be accessed selectively by the user. The server also returns the list of documents to user, and if the user is not logged on, the list will be returned to the user when he/she next logs on. It should be noted that the filter may be automatically created, based on the profile information or history information available on the interests of the user. Thus it would be possible, for example, for a server to determine interests of a user through earlier selections that were made.

The server receives the user's selections from the list and it updates the user's profile with the user's preferences. The server then returns the selected documents to user. As was described above, the knowledge engine server associates the search handle with user's selections and with the user's search strategy, storing that association in user's profile. It is understood that the term "document" under the present invention relates not just to web page documents, but also to services such as streaming video, audio or other application-specific communication. Thus, the present disclosure is not limited to just browsing after user discovery has taken place.

DESCRIPTION OF THE FIGURES:

Figure 1 is a network diagram of the invention, showing an examplary relationship between the user's Wireless Application Protocol (WAP)-enabled portable wireless device, the WAP protocol gateway to the Internet, the knowledge engine server, the Universal Description, Discovery and Integration (UDDI) registry, and a plurality of web sites;

Figure 1A through **1H** show the user's wireless device in a progression of stages as it carries out the UDDI registry browsing method;

Figure 1I and **1J** show the user's wireless device in a progression of stages as it carries out the method of replaying a UDDI registry search strategy as a shortcut for online or offline queries to the UDDI registry;

Figure 2 is a functional block diagram of the wireless device **100**, showing its various components and programs;

Figure 2A is a flow diagram of the sequence of operational steps for the wireless device's UDDI registry browsing program;

Figure 2B is a flow diagram of the sequence of operational steps for the wireless device's program to replay the UDDI registry search strategy;

Figure 3A is a network process flow diagram of the interaction of the wireless device and the UDDI registry when carrying out the UDDI registry browsing program of **Figure 2A**;

Figure 3B is a network process flow diagram of the interaction of the wireless device and the UDDI registry when carrying out the program to replay the UDDI registry search strategy;

Figure 4 is a functional block diagram of the knowledge engine server, showing the memory storing the application services software programs needed to perform the operations of the invention;

Figure 4A is a more detailed functional block diagram of the server, showing the knowledge engine;

Figure 4B is a network process flow diagram of the interaction of the wireless device, the knowledge engine server, and the UDDI registry when carrying out the UDDI registry browsing program in the server.

DISCUSSION OF THE PREFERRED EMBODIMENT:

The invention is applied to wireless telephones and wireless personal digital assistants (PDAs) implementing the Wireless Application Protocol (WAP) standard. **Figure 1** is a network diagram of an embodiment of the invention, illustrating a relationship between the user's Wireless Application Protocol (WAP)-enabled portable wireless device **100**, a WAP protocol gateway **120**, and the server **140**. The user's WAP-enabled portable wireless device **100** can be a wireless mobile phone, pager, two-way radio, smartphone, personal communicator, or the like. The user's WAP-enabled portable wireless device **100** accesses a small file called a deck which is composed of several smaller pages called cards which are small enough to fit into the display area of the device's microbrowser **102**. The small size of the microbrowser **102** and the small file sizes accommodate the low memory constraints of the portable wireless device **100** and the low-bandwidth constraints of a wireless network **116**. The cards are written in the Wireless Markup Language (WML) which is specifically devised for small screens and one-hand navigation without a keyboard. The WML language is scaleable from two-line text displays on the microbrowser **102** of a cellular telephone, up through large LCD screens found on smart phones and personal communicators. The cards written in the WML language can include programs written in WMLScript, which is similar to JavaScript, but makes minimal demands on memory

and CPU power of the device **100** because it does not contain many of the unnecessary functions found in other scripting languages.

The Nokia WAP Client Version 2.0 is a software product containing the components necessary to implement a WAP client on a wireless device. These components include a Wireless Markup Language (WML) Browser, WMLScript engine, Push Subsystem, and Wireless Protocol Stack. The Nokia WAP Client is a source-code product that can port and integrate into wireless devices such as mobile phones and wireless PDAs. Application programs stored in the wireless device interact with the WAP Client to implement a variety of communications applications. Details of the Nokia WAP Client Version 2.0 can be found in the online paper: Nokia WAP Client Version 2.0, Product Overview, Nokia Internet Communications, 2000, www.nokia.com/corporate/wap. It is also understood that the system and methods disclosed herein are applicable to other platforms as well, such as XHTML, and is not limited strictly to the WAP protocol.

The user's WAP-enabled portable wireless device **100** communicates with the radio tower **114** and can exchange messages for distances up to several kilometers. The types of wireless networks **116** supported by the WAP standard include Cellular Digital Packet Data (CDPD), Code-Division Multiple Access (CDMA), Global System for Mobile Communications (GSM), Time Division Multiple Access (TDMA), GPRS, 3G-Broadband, and the like.

The overall process of communication between the user's WAP-enabled wireless device (the client) **100**, through the WAP protocol gateway **120**, to the server **140** resembles the way Web pages are served on the Internet using the HyperText Transfer Protocol (HTTP) or World Wide Web protocol:

[1] The user presses a key on the user's device **100** related to the Uniform Resource Locator (URL) of the server **140**.

[2] The user's device **100** sends the URL, via the radio tower **114** and the wireless network **116**, to the gateway **120** using WAP protocols.

[3] The gateway **120** translates the WAP request into an HTTP request and sends it over the Internet **130** to the server **140**, via Transmission Control Protocol/ Internet Protocol (TCP/IP) interfaces.

[4] The server **140** handles the request just like any other HTTP request received over the Internet. The server **140** either returns a WML deck or a HyperText Markup Language (HTML) page back to the gateway **120** using standard server programs written, for example in Common Gateway Interface (CGI) programs, Java servlets, or the like.

[5] The gateway **120** receives the response from the server **140** on behalf of the user's device **100**. If the response is an HTML page, it gets transcoded into WML if necessary. Then the WML and WMLScript coding is encoded into a byte code that is then sent to the user's device **100**.

[6] The user's device **100** receives the response in the WML byte code and displays the first card in the deck on the microbrowser **102** to the user.

In **Figure 1**, the protocol gateway **120** includes a WAP protocol stack organized into five different layers (not shown). An application layer is the wireless application environment, which executes portable applications and services. A session layer is the wireless session protocol, which supplies methods for the organized exchange of content between client/server applications. A transaction layer is the wireless transaction protocol, which provides methods for

performing reliable transactions. A security layer is the wireless transport layer security, which provides authentication, privacy, and secure connections between applications. The transport layer is the wireless datagram protocol, which shelters the upper layers from the unique requirements of the diverse wireless network protocols, such as CDPD, CDMA, GSM, etc. Additional information about the WAP standard and the WAP protocol stack can be found in the book by Charles Arehart, et al. entitled, "Professional WAP", published by Wrox Press Ltd., 2000 (ISBN 1-861004-04-1).

In **Figure 1**, the user's portable wireless device **100** includes the microbrowser **102** displaying the Mobile Web Services menu, that enables the user to navigate through the cards being displayed and to select options that are programmed by the application programs **106**. The user's device **100** also includes the WAP client program **108** which has been previously discussed.

The Mobile Web Services menu displayed by the microbrowser **102** in **Figure 1** is rendered by the WAP client program **108** under the control of the application programs **106**, which are further illustrated in **Figures 2**, **2A**, and **2B**. The user can select the session type utilizing the Mobile Web Services menu, by either [A] a direct session with the UDDI registry or [B] an indirect session with the UDDI registry through a knowledge server **140**. The direct session with the UDDI registry is further illustrated in the network process flow diagrams of **Figures 3A** and **3B**. The knowledge service adds value to UDDI searches with pre and post-processing, regardless of the device being used to accesss UDDI. The indirect session with the UDDI registry through the knowledge server **140** is further illustrated in the network process flow diagram of **Figure 4B**. Whichever session type is selected by the user, the Mobile Web

Services menu of **Figure 1** presents to the user the UDDI Registry Search Menu from which the user can select the following options:

[1] BROWSE UDDI REGISTRY FOR WEB SITE URLS

[2] DRILL-DOWN UDDI DATA FOR DETAILS
- [2A] BUSINESS ENTITY DATA
- [2B] BUSINESS SERVICE DATA
- [2C] BINDING TEMPLATE DATA
- [2D] T_MODEL DATA

[3] INVOKE WEB SITE W/ CACHED UDDI DATA

[4] ENTER SEARCH HANDLE TO USE
PROFILE FROM A PRIOR SEARCH

Option [1] of browsing the UDDI registry for web site URLs allows the user to explore and examine data organized by the UDDI registry in a hierarchy. The core information model used by the UDDI registries is defined in an XML schema. XML allows hierarchical relationships to be described for four types data: business information; service information, binding information; and information about specifications for services.

A first type of data is Business information, which is specified in the UDDI registry with the *businessEntity* XML element. The *businessEntity* XML element typically includes the name, general description, contacts, and categories of the business whose web site is on the Web. The *businessEntity* XML element serves as the top of the information hierarchy for all of the information about a business under the present embodiment.

A second type of data is Service information, which is specified in the UDDI registry with the *businessService* XML element. One or more *businessService* XML elements is contained in each *businessEntity* XML element. The *businessService* XML element includes one

or more *bindingTemplate* XML elements, which are a third type of data. The *businessService* XML element is a descriptive container that is used to group a series of related Web services related to either a business process or category of services, such as purchasing services, shipping services, news services, and other high-level business processes. The *businessService* XML element information sets can each be further categorized in combinations of industry, product and service or geographic subjects.

The *bindingTemplate* XML element contains the detailed technical descriptions of Web services. As such, these structures provide the technical entry point URL for specific services and products offered by a business. Each *bindingTemplate* XML element structure has a single logical *businessService* XML element parent, which in turn has a single logical *businessEntity* XML element parent. An important piece of information in the *bindingTemplate* XML element is the *accessPoint* element. The *accessPoint* element is the URL of a specific service provided by the business. A single attribute is typically provided, and is identified in the present embodiment as *URLType*. The purpose of the *URLType* attribute is to facilitate searching for entry points associated with a particular type of service. An example might be a purchase order service that provides three entry points, one for HTTP, one for SMTP, and one for FAX ordering. In this example, a *businessService* XML element contains three *bindingTemplate* XML element entries, each with identical data with the exception of the *accessPoint* value and *URLType* value.

A fourth type of data in the UDDI registry is the *tModel* XML element, which is pointed to by a pointer in the *bindingTemplate* XML element. The *tModel* XML element specifies the protocols, interchange formats and interchange sequencing rules for accessing web pages from the business' server having the service information specified in the *businessService* XML element.

Option [1] of the Mobile Web Services menu of **Figure 1**, is:

[1] BROWSE UDDI REGISTRY FOR WEB SITE URLS

Option [1] of browsing the UDDI registry for web site URLs involves starting with some broad information, performing a search, finding general result sets and then selecting more specific information for drill-down. The UDDI registry accommodates the browse pattern with the *find_xx* API call. These calls form the search capabilities provided by the UDDI registry and are matched with return messages that return overview information to match the supplied search criteria. A typical browse sequence may involve finding whether a particular business has any information registered in the UDDI registry. This sequence would start with a call *find_business*, and may subsequently pass the first few characters of the businesses name. The UDDI registry would then return a *businessList* result. The *businessList* result is a list of overview information (keys, names and description) of the *businessEntity* information that matched the search results returned by the *find_business* query. **Figure 1A** through **1H** illustrate the user's wireless device in a progression of stages as it carries out the UDDI registry browsing method. **Figure 2A** illustrates a flow diagram of the sequence of operational steps for the wireless device's UDDI registry browsing program. **Figure 3A** illustrates a network process flow diagram, showing the interaction of the wireless device and the UDDI registry when carrying out the UDDI registry browsing program of **Figure 2A**.

Option [2] of the Mobile Web Services menu of **Figure 1**, is:

[2] DRILL-DOWN UDDI DATA FOR DETAILS

[2A] BUSINESS ENTITY DATA
[2B] BUSINESS SERVICE DATA
[2C] BINDING TEMPLATE DATA
[2D] T_MODEL DATA

After the user has identified a business he/she has been browsing for in Option [1], the user can drill-down into their *businessService* information. Here, the user may search for particular service types (e.g. purchasing, shipping, news and the like) using the *find_service* API call. Similarly, if the user knows the technical fingerprint (*tModel* signature) of a particular product and wants to see if the business he/she has chosen supports a compatible service interface, the user can use the *find-binding* API call.

Once the user has a key for one of the four main data types managed by the UDDI registry, he/she can use that key to access the full registered details for a specific data type (*businessEntity*, *businessService*, *bindingTemplate*, or *tModel*). The full registered information for any of these structures can be accessed by passing a relevant key type to one of the *get_xx* API calls to the UDDI registry.

Continuing with the example on browsing, one of the data items returned by all of the *findL-xx* return sets is key information. In the case of a business that the user is interested in, the *businessKey* value returned within the contents of a *businessList* structure can be passed as an argument to the UDDI registry to *get_businessDetail*. The successful return of this message is a *businessDetail* message containing the full registered information for the entity whose key value was passed. This typically will be a full *businessEntity* structure. Since full *businessEntity* structures can contain a large quantity of information, it can be optionally cached in the cache **144** of the knowledge engine server **140** of **Figure 1**.

Option [3] of the Mobile Web Services menu of **Figure 1**, is:

[3] INVOKE WEB SITE W/ CACHED UDDI DATA

After the user has retrieved the *businessEntity* structure from the UDDI registry **170** of **Figure 1**, the user can access the *accessPoint* element of the *bindingTemplate* XML element in the *businessEntity* structure to obtain the URL of a specific service provided by the business. Option [3] invokes the business' web site **160** with the cached URL to access the desired web pages. Since the accessed web pages from the web site **160** can contain a large quantity of information, it can be optionally cached in the cache **144** of the knowledge engine server **140** of **Figure 1**. Typically, if *bindingTemplet* is used to attempt contact services directly at the *accessPoint* and fails, the terminal may use the *bindingTemplate's* unique ID to fetch a new *bindingTemplate* instance from te UDDI registry, assuming that the new instance contains up-to-date information on the service.

Option [4] of the Mobile Web Services menu of **Figure 1**, is:

[4] ENTER SEARCH HANDLE TO USE PROFILE FROM A PRIOR SEARCH

Option [4] enables the user to replay a prior UDDI registry search strategy. The user typically enters a previously used search handle into the wireless device and selects the replay Option [4]. The wireless device then accesses the UDDI registry search strategy from the user's stored profile corresponding to the search handle and loads the business name query, the selected *businessEntity* data, the selected *businessService* data, and the selected *bindingTemplate* data as each respective operand that would have been otherwise entered by the user. If the data in the

UDDI registry has been updated since the user's last query, the bindingTemplate data may contain additional or modified *accessPoint URLs*, of the selected service on the web site of the selected business. **Figure 1I** and **1J** illustrate the user's wireless device in a progression of stages as it carries out the method of replaying a UDDI registry search strategy as a shortcut for online or offline queries to the UDDI registry. **Figure 2B** discloses a flow diagram of the sequence of operational steps for the wireless device's program to replay the UDDI registry search strategy. **Figure 3B** is a network process flow diagram of the interaction of the wireless device and the UDDI registry when carrying out the program to replay the UDDI registry search strategy.

The sequence of operational steps for the wireless device's UDDI registry browsing method are shown in **Figure 2A**. The Mobile Web Services menu of **Figure 1A** begins by entering a search handle that will be associated with the user's search strategy. The query terms are subsequently entered by the user, which may be a business name, part of a business name, a service description, or other characterization. If the characterization is part of a business name, the wireless device then sends a *find_business* XML inquiry using Simple Object Access Protocol (SOAP) to the UDDI registry. The wireless device then receives back from the UDDI registry, a *businessList* message shown in the Mobile Web Services menu of **Figure 1B**, that contains a list of business names satisfying the *find_business* query. The user can then select an item from the returned *businessList* message and drill-down in the selected business' entity data.

The Mobile Web Services menu of **Figure 1C** shows the wireless device then sending a *find_service* XML inquiry using SOAP to the UDDI registry. The Mobile Web Services menu of **Figure 1D** shows the wireless device then receiving back from the UDDI registry, a *serviceList* message that contains a list of names of services offered by the selected business. The user can

then select an item from the returned *serviceList* message and drill-down in the selected service data, as shown by the Mobile Web Services menu of **Figure 1E**.

The wireless device then sends a *_get_serviceDetail_* XML inquiry using SOAP to the UDDI registry, as shown by the Mobile Web Services menu of **Figure 1E**. The Mobile Web Services menu of **Figure 1F** then shows the wireless device receiving back from the UDDI registry a *serviceDetail* message that includes bindingTemplate data that contains the details of the selected service. Included in the *bindingTemplate* data is the *accessPoint URL*, which is the URL of the selected service on the web site of the selected business.

The Mobile Web Services menu of **Figure 1G** shows the wireless device displaying the accessPoint URL to the user. The Mobile Web Services menu of **Figure 1H** shows that the *serviceDetail* message can contain one or a plurality of URLs, depending on the number of matches made against the user's query in the search by the UDDI registry. The wireless device also stores the search handle in user profile with the selected accessPoint URL, to enable the user to access web pages from the web site of the selected business. This provides the user with a shortcut for accessing pages from web sites, in response to the user's entry of abbreviated search handle to the wireless device.

The wireless device also stores the search handle in user profile with the UDDI registry search strategy. The stored search strategy includes the business name query , the selected *businessEntity* data, the selected *businessService* data, the selected *bindingTemplate* data, and the selected *accessPoint* URL. This provides the user with a shortcut for online or offline queries to the UDDI registry, in response to the user's entry of abbreviated search handle to the wireless device.

To replay a UDDI registry search strategy, the user enters a search handle into the wireless device and selects the replay option, as shown in the Mobile Web Services menu of **Figure 1I**. The wireless device then accesses the UDDI registry search strategy from user profile corresponding to the search handle and loads the business name query , the selected *businessEntity* data, the selected *businessService* data, and the selected *bindingTemplate* data as each respective operand that would have been otherwise entered by the user. If the data in the UDDI registry has been updated since the user's last query, the bindingTemplate data may contain additional or modified *accessPoint URLs*, of the selected service on the web site of the selected business, as shown by the Mobile Web Services menu of **Figure 1J**.

Figure 2 is a functional block diagram of the wireless device **100**, and illustrates its various components and programs. **Figure 2** discloses the memory **202** connected by means of the bus **204** to the radio **206**, the keypad **104**, the central processor **210** and the display **212**. The memory **202** contains program modules each consisting of a sequence of programmable instructions. The wireless devices UDDI registry browsing program **240** (see **Figure 2A**) is stored in the memory **202**. The wireless devices replay UDDI registry search strategy program **270** (see **Figure 2B**) is also stored in the memory **202**. The indirect session thru server program **220** is also stored in the memory **202**.

User data **222** is stored in the memory **202**, which includes the user ID **230** and the user profile **232**. The user profile **232** includes the user's name and email address, the user's search handles, the UDDI search strategies, the sorting and filtering specifications, selected URLS, selected document titles and binding templates which contain URLS. Also contained in the memory **202** of **Figure 2** is the cache **224** wherein documents and lists returned from a website, can be stored. In addition, the WAP client program **108** is stored in the memory **202**.

Figure 2a is a flow diagram disclosing the sequence of operational steps for the wireless device's UDDI registry browsing program **240**. The steps depicted in **Figure 2A** are as follows:

Step 250: WIRELESS DEVICE BROWSING UDDI REGISTRY ENTER SEARCH HANDLE "_STOCKS_"

Step 252: ENTER QUERY TERMS "_WALL STREET_" + "_FINANC*_" IS THIS A BUSINESS NAME? Y/N "_Y_" IS THIS A SERVICE NAME? Y/N "_N_"

Step 254: SEND "_find_business_" XML INQUIRY WITH SOAP PROTOCOL TO UDDI REGISTRY

Step 256: SELECT ITEM FROM RETURNED businessList MESSAGE DRILL-DOWN BUSINESS ENTITY DATA "_WALL STREET JOURNAL ONLINE_"

Step 258: SEND "_find_service_" XML INQUIRY WITH SOAP PROTOCOL TO UDDI REGISTRY

Step 260: SELECT ITEM FROM RETURNED serviceList MESSAGE DRILL-DOWN BUSINESS SERVICE DATA "_TECH STOCK NEWS_"

Step 262: SEND *_get_serviceDetail_* XML INQUIRY WITH SOAP PROTOCOL TO UDDI REGISTRY

Step 264: SELECT ITEM OF RETURNED serviceDetail MESSAGE DISPLAY accessPoint URL FROM bindingTemplate DATA "COMPUTERS" IN RETURNED serviceDetail MESSAGE URL = "_www.wsj.com\news\computers_"

Step 266: STORE SEARCH HANDLE "_STOCKS_" IN USER PROFILE WITH SELECTED URL = "_www.wsj.com\news\computers_"

Step 268: STORE SEARCH HANDLE "_STOCKS_" IN USER PROFILE WITH UDDI REGISTRY SEARCH STRATEGY: BUSINESS NAME QUERY "_WALL STREET_" + "_FINANC*_" businessEntity DATA SELECTED "_WALL STREET JOURNAL ONLINE_" businessService DATA SELECTED "_TECH STOCK NEWS_" bindingTemplate DATA SELECTED "_COMPUTERS_" accessPoint URL SELECTED "_www.wsj.com\news\computers_"

Figure 2B shows a flow diagram of the sequence of operational steps for the wireless device's program to replay the UDDI registry search strategy. **Figure 2B** includes the following steps:

Step 271: REPLAY UDDI REGISTRY SEARCH STRATEGY ENTER SEARCH HANDLE "_STOCKS_".

Step 272: ACCESS UDDI REGISTRY SEARCH STRATEGY IN USER PROFILE FOR SEARCH HANDLE "_STOCKS_", BUSINESS NAME QUERY "_WALL STREET_" + "_FINANC_", businessEntity DATA SELECTED "_WALL STREET JOURNAL ONLINE_", businessService DATA SELECTED "_TECH STOCK NEWS_", bindingTemplate DATA SELECTED "_COMPUTERS_", accessPoint URL SELECTED "_www.wsj.com\news\computers_".

Step 274: SEND *_find_business_* SML INQUIRY WITH SOAP PROTOCOL TO UDDI REGISTRY.

Step 276: SELECT ITEM FROM RETURNED businessList MESSAGE DRILL-DOWN BUSINESS ENTITY DATA "_WALL STREET JOURNAL ONLINE_".

Step 278: SEND *_find_service_* XML INQUIRTY WITH SOAP PROTOCOL TO UDDI REGISTRY.

Step 280: SELECT ITEM FROM RETURNED serviceList MESSAGE DRILL-DOWN BUSINESS SERVICE DATA "_TECH STOCK NEWS_".

Step 282: SEND *_get_serviceDetail_* XML INQUIRY WITHSOAP PROTOCOL TO UDDI REGISTRY.

Step 284: SELECT ITEM OF RETURNED serviceDetail MESSAGE DISPLAY accesPoint URLs FROM bindingTemplate DATA "COMPUTERS" IN RETURNED serviceDetail MESSAGE, URL = "_www.wsj.com\news\computers_", URL = "_www.wsj.com\quotes\computers_", URL = "_www.wsj.com\graphs\computers_".

Figure 3A discloses a network process flow diagram showing the interaction of the wireless device and the UDDI registry when carrying out the UDDI registry browsing program of **Figure 2A**. The network process flow diagram in **Figure 3A** consists of three columns labeled respectively, wireless device **100**, server **140** and UDDI registry **170**. The process flow diagram **Figure 3A** illustrates the interaction between steps carried in the wireless device **100** and steps carried out in the UDDI registry **170**. The diagram of **Figure 3A** begins with the wireless device **100** step **250** where a search handle is entered. Then, at step **252**, query terms are entered. At step **254**, the *_find_business_* query is sent to the UDDI registry **170**. In the UDDI registry column **170** in **Figure 3A**, the UDDI registry conducts searches based on the *_find_business_* query and returns the *businessList* in step **255**. The flow then returns to the wireless device **100** and passes to step **256** wherein an item of the *businessList* is selected which typically is a *businessEntity*. The *businessEntity* is then drilled down. The flow then passes to step **258** in which the *_find_service_* query is sent to the UDDI registry. The flow then passes to the UDDI registry **170** where the *_find_service_* query is searched and the service list is returned in step **259**. Then the flow passes to the wireless device **100** where an item of the *serviceList* is selected which is a *businessService*. The *businessService* is subsequently drilled down in step **260**. The flow proceeds to step **262** in which the *_get_serviceDetail_* query is sent to the UDDI registry. Then the flow passes to the UDDI registry **170** where the *_get_serviceDetail_* query is responded to and the binding template is returned in step **263**. Then the flow passes back to the wireless device **100** where in step **264**, the *serviceDetail* is selected from the *bindingTemplate* and the accessPoint URL is stored in step **264**.

Figure 3B illustrates a network process flow diagram of the interaction of the wireless device and the UDDI registry when carrying out a program to replay the UDDI registry search

strategy. **Figure 3B** is divided into three columns, the wireless device column **100**, the server **140** column, and the UDDI registry **170** column. **Figure 3B** starts with the wireless device entering the replay UDDI registry search strategy wherein the search handle is entered in step **271**. Then the process flows to step **272** where the search strategy is accessed in the user profile which corresponds to the search handle which was input in step **271**. **Figure 3B** then proceeds through the remainder of the processes in a similar manner as that disclosed in **Figure 3A**. **Figure 3B** discloses how the user is enabled to replay a prior UDDI registry search strategy. The user merely enters a previously used search handle into the wireless device and selects the replay option. The wireless device then accesses the UDDI registry search strategy from the user's stored profile corresponding to that search handle. This may be performed either at the wireless device **100** or, in the alternate embodiment in the server **140**. The search strategy includes information such as the *businessEntity* data and *businessService* data and *bindingTemplate* data that was found during the course of an earlier search of the UDDI registry **170**. This data contained in the UDDI registry search strategy (accessed from the stored profile) is then loaded as the data for each respective operand that would have been otherwise entered by the user. In this way, the flow diagram of **Figure 3B** enables the user to automatically invoke a search strategy of the UDDI registry **170**, that was used in the past.

Figure 4 is a functional block diagram of the knowledge engine server, showing the memory storing the application services software programs needed to perform the operations of an embodiment of the invention. **Figure 4** discloses the functional components of an exemplary knowledge engine server **140** arranged as an object model. The object model groups the object oriented software programs into components that perform the major functions and appliations in knowledge engine server **140**. The object model for memory **402** of knowledge engine server

140 employs a three-tier architecture that includes presentation tier **415**, infrastructure objects partition **422**, and business logic tier **414**. The object model further divides business logic tier **414** into two partitions, application objects partition **422** and data objects partition **426**.

Presentation tier **415** retains the programs that manage the device interfaces to knowledge engine server **140**. In **Figure 4**, presentation tier **415** includes network interface **420**. A suitable implementation of presentation tier **415** may use Java servlets to interact with WAP protocol gateway **120** via the hypertext transfer protocol ("HTTP"). The Java servlets run within a request/response server that manages the exchange of messages between WAP protocol gateway **120** and knowledge engine server **140**. A Java servlet is a Java program that runs within a Web server environment. A Java servlet takes a request as input, parses the data, performs logic operations, and issues a response back to WAP protocol gateway **120**. The Java runtime platform pools the Java servlets to simultaneously service many requests. Network interface **420** accepts request messages from WAP protocol gateway **120** and passes the information in the request to visit object **428** for further processing. Visit object **428** passes the result of that processing to network interface **420** for transmission back to the WAP protocol gateway **120**. Network interface **420** may also use network adapter **406** to excahnge data with another user device.

Infrastructure objects partition **422** retains the programs that perform administrative and system functions on behalf of business logic tier **414**. Infrastructure objects partition **422** includes operating system **425**, and an object oriented software program component for database server interface **430**, and system administrator interface **432**.

Business logic tier **414** in **Figure 4** includes multiple instances of visit object **428**, **428'**, **428"**. A separate instance of visit object **428** exists for each network interface **420** session. Each

visit object **428** is a stateful session bean that includes a persistent storage area from initiation through termination of the session, not just during a single interaction or method call. The persistent storage area retains information associated with the session.

When WAP protocol gateway **120** sends a query message to knowledge engine server **140**, the message is sent to network interface **420** to invoke a method that creates visit object **428** and stores connection information as a state in visit object **428**. Visit object **428** may, in turn, invoke a method in UDDI registry browsing application **440** to browse the UDDI registry **170**. Application **440** sends queries to the UDDI registry, as discussed above.

When WAP protocol gateway **120** sends a search handle message to knowledge engine server **140**, a message is sent to network interface **420** to invoke a method that creates visit object **428** and stores connection information as a state in visit object **428**. Visit object **428** may, in turn, invoke a method in replay UDDI registry search strategy application **442** to replay a prior search strategy. The application **442** performs the replay method discussed above. Similar operations occur for applications **444**, **446** and **448** in **Figure 4**.

Figure 4A is a more detailed functional block diagram of the server, showing the knowledge engine **142**. The knowledge engine **142** includes a program which is shown **in Figure 4A** as a sequence of steps 1 thru 13.

KNOWLEDGE ENGINE **142**

[1] RECEIVE USER'S QUERY OR SEARCH HANDLE
[2] IF SEARCH HANDLE RECEIVED, THEN REPLAY UDDI SEARCH PROFILE & GOTO 6
[3] ACCESS UDDI REGISTRY WITH USER'S QUERY TO IDENTIFY WEB SITES
[4] RETURN LIST OF WEB SITES TO USER AND STORE BINDING TEMPLATES IN PROFILE

[5] RECEIVE USER'S SELECTION OF WEB SITES TO SEARCH & UPDATE USER PROFILE
[6] SEARCH THE IDENTIFIED WEB SITES USING URLS FROM BINDING TEMPLATES
[7] RETRIEVE DOCUMENTS RESULTING FROM SEARCH
[8] SORT LIST OF DOCUMENTS IN ACCORDANCE WITH USER'S PROFILE
[9] STORE DOCUMENTS AND LIST IN CACHE
[10] RETURN LIST OF DOCUMENTS TO USER WHENEVER USER IS LOGGED ON
[11] RECEIVE USER'S SELECTIONS FROM LIST AND UPDATE USER'S PROFILE
[12] RETURN SELECTED DOCUMENTS TO USER
[13] ASSOCIATE SEARCH HANDLE WITH USER'S SELECTIONS IN USER'S PROFILE

Also provided in server **140** is the user data **146** which includes the user ID profile **230** which is discussed above. Since a plurality of users may make use of the server **140**, there are a plurality of user profiles shown in **Figure 4A**, one for the user ID **230'** having user profile **232'** and another for the user ID **230"** having user profile **232"**. The server **140** of **Figure 4A** also has a cache **144** which stores documents and lists which are obtained from the various websites **160** that have been interrogated by the user with the aid of the server **140**.

Figure 4B is a flow diagram of the sequence of operational steps for the server **140** UDDI registry browsing program **170**. **Figure 4B** has three columns, the first column labeled user's wireless device **100**, the second column labeled server **140**, and a third column labeled UDDI registry **170** and web sites **160**. **Figure 4B** illustrates the interaction of the wireless device **100**, the server **140**, the UDDI registry **170** and the web sites **160**, in accordance with an embodiment of the invention. Starting with the user's wireless device **100**, **Figure 4B** shows sending a query to the server, in step **302**. At the server **140**, the query is received from the user in step **304**, and the process flows to step **306** where web sites are identified from the UDDI registry and the user's profile is updated. The process in step **306** for identification of the web sites from the

UDDI registry is the process which has been discussed above in connection with **Figures 2A** and **3A**. The process then flows to step **330** in the UDDI registry **170**, wherein the UDDI registry accesses the requested information in response to the queries sent from the server **140** to identify web sites. Step **330** then transfers the results of that search from the UDDI registry **170** back to the server **140**. At the server **140**, the process flows to step **314** wherein the server has taken the information identifying the web sites received from the UDDI registry **170**, and formulates a request to retrieve documents which is sent to the web sites **160**. The process then flows to step **332** where the web sites **160** receiving the request from the server **140**, access their respective severs for the requested documents and then return the documents to the server **140**. The server **140** then sorts the documents into a list in accord with the user's profile, sorting the list into the order requested by the user, and filtering out any documents which the user is not interested, in accordance with the user's profile. The process then flows to step **316** in which the documents are stored in the cache at the server, cache **144**, and the list which has been sorted by the server **140**, is returned to the user. The process then flows to step **320** at the user's wireless device **100** where the sorted list received from the server **140** is presented to the user and the user can select from that list those documents desired to be reviewed. In step **320**, the user's request for documents is then sent back to the server **140**. The process then flows to step **326** where the server **140** accesses its cache **144** to retrieve those documents selected by the user in step **320**, and then the server **140** returns the selected documents to the user's wireless device **100**. Step **326** then compiles the user's preferences in the user profile **230**. The server **140** can also update the user's preferences in the user's profile in step **328**. The process flows from step **326** to step **322** at the user's wireless device **100**, where the user receives the selected documents.

Although illustrative embodiments have been described herein in detail, it should be noted and understood that the descriptions and drawings have been provided for purposes of illustration only and that other variations both in form and detail can be made thereupon without departing from the spirit and scope of the invention. The terms and expressions have been used as terms of description and not terms of limitation. There is no limitation to use the terms or expressions to exclude any equivalents of features shown and described or portions thereof.